REMARKS

Claims 1-3, 8-12, and 21 are pending. Claim 1 stands rejected under 35 U.S.C. § 112, ¶ 2 as being indefinite for failing to particularly point and distinctly claim the subject matter which Applicant regards as the invention. Claims 1-3, 8-12, and 21 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 5,778,211 to Hohensee et al. in view of U.S. Patent No. 5,915,117 to Ross et al. Claims 1-3, 8-12, and 21 stand rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1-19 of U.S. Patent No. 6,173,248.

Reconsideration is requested. No new matter is added. The rejections are traversed. Claims 1-3, 8-12, and 21 remain in the case for consideration.

DOUBLE PATENTING REJECTION

A terminal disclaimer will be filed to overcome the obviousness-type double patenting rejection over U.S. Patent No. 6,173,248, once claims 1-3, 8-12, and 21 are indicated as otherwise allowable.

REJECTIONS UNDER 35 U.S.C. § 112, ¶ 2

The Examiner rejected claim 1 as being indefinite for failing to particularly point out and distinctly claim the subject matter which the Applicant regards as the invention. In particular, the Examiner indicated that claim 1 is incomplete because claim 1 does not explain how to handle the exception if the exception was determined to be caused by the user program.

The Applicant does not believe claim 1 is indefinite. As written, claim 1 explains how the claim is handled when the exception is determined to be caused by the emulator program. The Applicant believes that this claim is patentable by itself, without any limitation on how the exception is handled if it is determined to be caused by the user program. In the invention of claim 1, if the exception is caused by the user program, it can be handled in any manner determined to be appropriate. For example, it is known in the art to abort a program when the program causes an exception. Or the exception can be delivered to the user program, to be handled in accordance with the exception handler built into the user program. (This latter approach is consistent with the invention as defined in claim 2.) A person skilled in the art will recognize that other techniques could be used to handle the exception where the exception is determined to be caused by the user program.

Because the invention as defined in claim 1 does not require a description about how to handle an exception caused by the user program, claim 1 is allowable over 35 U.S.C. § 112, ¶ 2.

REJECTIONS UNDER 35 U.S.C. § 103(a)

Referring to claim 1, the invention is directed toward a method for processing an exception in an emulator program. The exception is received from an operating system of a computer. Whether the exception was caused by the emulator program or the user program being emulated using the emulator program is determined. If the exception was caused by the emulator program, then the emulator program handles the exception internally, without delivering the exception to the user program.

Claim 2 elaborates on the invention of claim 1. In claim 2, if the exception was caused by the user program, then the type of the exception is determined. The emulator program determines whether the type of exception is currently blocked by the user program, and if the type of exception is not currently blocked, then the emulator program delivers a notification of the exception to the user program.

In contrast, Hohensee teaches a system and method for emulating a delayed exception in a computer using a precise exception mechanism. As defined in Hohensee, a delayed exception architecture is one where the handling of an exception is delayed (that is, not processed immediately) until it must be processed. As an example, Hohensee describes the processing of a floating-point instruction in a delayed exception architecture. If an exception is raised during the processing of the floating-point instruction, the exception is not processed immediately. Instead, processing is deferred until such time as another floating-point instruction is encountered. The rationale behind this approach is that the exception might not need to be handled, and so the exception is processed only when it is certain that the exception has to be handled. A precise exception architecture, on the other hand, operates on the principle that an exception should be handled immediately. This approach avoids the need to buffer the exception, but at the cost of processing an exception unnecessarily.

An emulator designed to emulate a delayed exception architecture running on a precise exception architecture needs to account for this difference in the timing of exception processing. This is the focus of the Hohensee patent. Hohensee teaches how to design and operate an emulator that emulates a delayed exception architecture. During emulation of the program, Hohensee teaches checking a pending exception indication state. If an exception is pending, then the exception is handled to move the emulator into a no-pending-exception indication state. The instruction to be performed is then emulated, and if an exception is detected while processing the instruction, the pending-exception indicator state is set.

The difference between the invention and Hohensee lies in the language of claim 1 that the Examiner referred to in the § 112, ¶ 2 rejection. Hohensee *assumes* that the emulator program runs completely without error, and so any exceptions raised are caused by the program being emulated. The invention of claim 1 makes no such assumption. Quite the contrary, the invention of claim 1 is concerned with the case in which the emulator program itself caused the exception – a situation Hohensee does not address. If the emulator program caused the exception, then the emulator program itself needs to handle the exception. But in that case, the user program was not at fault for causing the exception, and does not need to be notified about the exception. On the other hand, if the user program caused the exception (claim 2), then the emulator program can deliver notice of the exception to the user program (assuming the user program has not blocked the type of exception).

Hohensee does not teach the concept of determining whether the exception was caused by the emulator program or the user program. Instead, as indicated above, Hohensee assumes that the exception was caused by the user program. Further, Hohensee is only concerned with the timing of processing the exception. Accordingly, the invention and Hohensee can be combined: the invention can be enhanced by Hohensee in the case where the emulator program is emulating the execution of a program designed for a delayed exception architecture on a precise exception architecture. This would affect when the notification of the exception is delivered to the user program (assuming the user program has not blocked the exception type). But Hohensee does not make obvious determining whether the emulator program or the user program caused the exception. And if Hohensee does not teach the concept of the emulator program causing an exception, then Hohensee cannot teach how the emulator program can handle an exception without delivering a notification of the exception to the user program.

The Examiner referred to column 7, line 41, through column 8, line 4, and columns 13-14 as support that Hohensee teaches the concept of determining whether the exception was caused by the emulator program or the user program. But column 7, line 41, through column 8, line 4, only describes how an exception is processed in a precise exception architecture. And columns 13-14 describe in detail how a floating point exception occurring in a delayed exception architecture can be emulated using a precise exception architecture. No mention is made in the cited locations (or anywhere else) of distinguishing between exceptions caused by the emulator program and exceptions caused by the user program.

The invention as defined by claim 1 is directed toward:

> A method for processing an exception in an emulator program running on a digital computer having a memory and under control of an operating system, the emulator program emulating execution of a user program constructed for execution on a legacy platform, the method comprising the steps of:
>
> receiving an exception from the operating system during operation of the emulator program;
>
> *determining whether the received exception was caused by the emulator program itself or by the user program*; and
>
> *if the exception was caused by the emulator program, handling the exception internally in the emulator program without delivering the exception to the emulated user program.*

(claim 1; italics added). As these features are not taught or suggested by Hohensee, claim 1 is patentable under 35 U.S.C. § 103(a) over Hohensee. Accordingly, claims 1-3, 8-12, and 21 are allowable.

The invention as defined by claim 2 is directed toward:

> A method for processing an exception according to claim 1 and further comprising, *if the exception was caused by the user program*:
>
> identifying the type of exception;
>
> *determining whether the identified type of exception is currently blocked by the user program*; and
>
> *if the identified type of exception is not currently blocked by the user program, delivering notification of the exception to the user program.*

(claim 2; italics added). As these features are not taught or suggested by Hohensee, claim 2 is patentable under 35 U.S.C. § 103(a) over Hohensee. Accordingly, claims 2-3 and 8-12 are allowable.

Applicant respectfully submits that each of the Examiner's rejections has been overcome and that this Application is in condition for allowance. Such is respectfully requested. The Examiner is encouraged to telephone the undersigned at (503) 222-3613 if it appears that an interview would be helpful in advancing the case.

Please address all future communications to:

IP Administration
Legal Department, M/S 4059
HEWLETT-PACKARD COMPANY
11000 Wolfe Road
Cupertino, CA 95014-0691

Direct all telephone calls to:

David A. Plettner, Reg. No. 36,241
(408) 447-3013

Respectfully submitted,

MARGER JOHNSON & McCOLLOM, P.C.

[illegible]

Ariel S. Rogson
Reg. No. 43,054

MARGER JOHNSON & McCOLLOM, P.C.
1030 SW Morrison Street
Portland, OR 97205
Telephone (503) 222-3613
Facsimile (503) 274-4622
ariel@techlaw.com

I HEREBY CERTIFY THAT THIS CORRESPONDENCE IS BEING DEPOSITED WITH THE UNITED STATES POSTAL SERVICE AS FIRST CLASS MAIL IN AN ENVELOPE ADDRESSED TO:

☐ COMMISSIONER OF PATENTS AND TRADEMARKS WASHINGTON D.C. 20231

☒ MAIL STOP________
COMMISSIONER FOR PATENTS
BOX 1450
ALEXANDRIA, VA 22313-1450

☐ BOX ________
COMMISSIONER FOR TRADEMARKS 2900 CRYSTAL DRIVE ARLINGTON, VA 22202-3513

ON: 7 Aug 2003

[illegible]